第3章 关系型数据库的基础设计

学习目的与要求

本章主要介绍关系型数据库的基础设计。通过本章的学习，主要掌握关系型数据库设计流程、AscentWebDb 医药商务系统的数据库设计和 Power Designer 的使用。

本章主要内容

- 数据库的设计；
- 关系型数据的设计与优化；
- 数据的关系模型；
- 项目案例。

3.1 数据库设计

数据库设计是指根据用户的需求，在某一具体的数据库管理系统上，设计数据库的结构和建立数据库的过程，是一种基于用例驱动的设计过程。数据库设计是建立数据库应用系统的工作步骤中的关键的环节，是信息系统开发中的核心，由于数据库应用系统的复杂性，为了支持相关程序运行，数据库设计就变得异常复杂，因此最佳设计不可能一蹴而就，而只能是一种“反复探寻，逐步求精”的过程，也就是规划和结构化数据库中的数据对象以及这些数据对象之间关系的过程。

3.1.1 数据库设计流程

1. 需求分析

调查和分析用户的业务活动和数据的使用情况，弄清所用数据的种类、范围、数量以及它们在业务活动中交流的情况，确定用户对数据库系统的使用要求和各种约束条件

等，形成用户需求规约。

2. 概念设计

对用户要求描述的现实世界（可能是一件商品、一间商场或者一个人等），通过对其中住处的分类、聚集和概括，建立抽象的概念数据模型。这个概念模型应反映现实世界各部门的信息结构、信息流动情况、信息间的互相制约关系以及各部门对信息储存、查询和加工的要求等。所建立的模型应避开数据库在计算机上的具体实现细节，用一种抽象的形式表示出来。

1）概念模型的建模过程

运用概念目录列表或名词性短语找出问题领域中的备选概念。

绘制概念到概念模型图中。

为概念添加属性。

为概念添加关系。

2）概念模型设计

概念模型不依赖于具体的计算机系统，它是纯粹反映信息需求的概念结构。

建模是在需求分析结果的基础上展开的，常常要对数据进行抽象处理。常用的数据抽象方法是“聚集”和“概括”。

概念模型设计可分三步完成。

（1）设计局部概念模型

① 确定局部概念模型的范围。

② 定义实体。

③ 确定属性。

④ 定义联系。

⑤ 逐一画出所有的局部 E-R(Entity-Relationship)图，并附以相应的说明文件。

（2）设计全局概念模型

建立全局 E-R(Entity-Relationship)图的步骤如下：

① 确定公共实体类型。

② 合并局部 E-R(Entity-Relationship)图。

③ 消除不一致因素。

④ 优化全局 E-R(Entity-Relationship)图。

⑤ 画出全局 E-R(Entity-Relationship)图，并附以相应的说明文件。

关于 E-R(Entity-Relationship)图的介绍，请参考 3.1.2 节。

（3）概念模型评审

概念模型设计完成后，在进行下一步操作前，需要进行确认和评审，评审需要两部分角色参与，第一部分是用户评审，第二部分是开发人员评审。

3. 逻辑设计

主要工作是将现实世界的概念数据模型设计成数据库的一种逻辑模式，即适应于某种特定数据库管理系统所支持的逻辑数据模式。与此同时，可能还需要为各种数据处理应用领域产生相应的逻辑子模式。这一步设计的结果就是所谓的“逻辑数据库”。

逻辑设计主要包括以下内容：

(1) 分析业务实体域。在概念模型设计中，我们确定了几个基本的业务实体域，但是，数据库的设计方法是一个逐步求精的过程，在进行设计时，一般是一次一个业务实体或一次若干个业务实体地逐步完成的。所以，我们必须对概念模型设计步骤中确定的几个业务实体域进行分析，一并选择首先要实施的业务实体域。选择第一个业务实体域所要考虑的是它要足够大，以便使得该业务实体域能建设成为一个可应用的系统，它还要足够小，以便于开发和较快地实施。

(2) 粒度层次划分。所谓粒度，是设计数据库的一个最重要方面。

粒度是指数据库的数据单位中保存数据的细化或综合程度的级别。细化程度越高，粒度级就越小；相反，细化程度越低，粒度级就越大。数据的粒度一直是一个设计问题。在早期建立的操作型系统中，粒度是用于访问授权的。当详细的数据被更新时，几乎总是把它存放在最低粒度级上。但在数据库环境中，对粒度不做假设。在数据仓库环境中粒度之所以是主要的设计问题，是因为它深深地影响存放在数据库中的数据量的大小，同时影响数据仓库所能回答的查询类型。在数据仓库中的数据量大小与查询的详细程度之间要做出权衡。

由于业务实体数据库响应企业级业务需求，所以必须保存最细粒度数据，同时根据业务部门的查询需求考虑确定多重粒度来提高复杂查询速度。

(3) 确定数据分割策略。选择适当的数据分割的标准，一般要考虑以下几方面因素：数据量(而非记录行数)、数据分析处理的实际情况、简单易行以及粒度划分策略等。其中，数据量的大小是决定是否进行数据分割和如何分割的主要因素；数据分析处理的要求是选择数据分割标准的一个主要依据，因为数据分割是跟数据分析处理的业务实体紧密联系的。

(4) 关系模式定义。在概念模型设计时，我们就确定了数据仓库的基本实体，并对每个业务实体的公共码键、基本内容等做了描述。这里，我们将要对选定的当前实施的业务实体进行模式划分，形成多个表，并确定各个表的关系模式。

4. 物理设计

根据特定数据库管理系统所提供的多种存储结构和存取方法等依赖于具体计算机结构的各项物理设计措施，对具体的应用任务选定最合适的物理存储结构(包括文件类型、索引结构和数据的存放次序与位逻辑等)、存取方法和存取路径等。这一步设计的结果就是所谓“物理数据库”。

5. 验证设计

在上述设计的基础上，收集数据并具体建立一个数据库，运行一些典型的应用任务来验证数据库设计的正确性和合理性。一般地，一个大型数据库的设计过程往往需要经过多次循环反复。当设计的某一步发现问题时，可能就需要返回到前面去进行修改。因此，在做上述数据库设计时就应考虑到今后修改设计的可能性和方便性。

6. 运行与维护设计

在数据库系统正式投入运行的过程中，必须不断地对其进行评估、调整与修改。

至今，数据库设计的很多工作仍需要人工来做，除了关系型数据库已有一套较完整的数据范式理论可用来部分地指导数据库设计之外，尚缺乏一套完善的数据库设计理论、方法和工具，以实现数据库设计的自动化或交互式的半自动化设计。所以数据库设计今后的研究发展方向是研究数据库设计理论，寻求能够更有效地表达语义关系的数据模型，为各阶段的设计提供自动或半自动的设计工具和集成化的开发环境，使数据库的设计更加工程化、更加规范化和更加方便易行，使得在数据库的设计中充分体现软件工程的先进思想和方法。

3.1.2 E-R(Entity-Relationship)图的概念

1. 什么是实体

实体(Entity)——是亚里士多德首创的一个重要哲学概念，也是后来西方哲学史上许多哲学家使用的重要哲学范畴，又译为本体。其含义一般是指能够独立存在的、作为一切属性的基础和万物本原的东西。亚里士多德认为实体是独立存在的东西，是一切属性的承担者，从语言和逻辑上说，它处于主语地位，其他表示数量、性质的范畴需要依附于实体，处于宾语的地位，只能用来说明主语。

亚里士多德认为实体的主要特征如下：

(1) 实体是独立的，是特定与某一类事物的并且可以与其他事物分离存在的。

(2) 实体在保持自身不变的同时，允许“由于自身变化”而产生不同的性质或者表现，例如，同一个学生可以有时成绩好，有时成绩不好，有时喜欢调皮捣蛋，但仍是这一个人，而不是其他的事物，那么学生就是实体。

(3) 实体是变中不变的东西，是生成变化的基础，是一组具有相同属性事物的集合，对此，亚里士多德明确指出：“这是实体的最突出的标志”。

总之，实体是客观事物或者逻辑事物的抽象，是可以独立存在的，由构成该事物的其他属性组成的一个概念，是可以包含多种变化的一个名词性的事物。

2. 什么是属性

属性即事物本身所固有的性质，是物质必然的、基本的、不可分离的特性，又是事物某个方面质的表现。一定质的事物常表现出多种属性。

通常说的一个属性是指属于某种实体的，用于描述该实体的特定的一种性质，如“人”这个实体有肤色，有性别，肤色和性别都是用来描述人的不同的两个性质，但是属于“人”这个实体。因此，属性是描述某种事物所具有的性质，不能独立存在，必须隶属于或包含于某种实体。

因此，实体是一种属性的集合概念，而该集合中的元素则是用于描述该实体不同性质特征的属性。

3. 什么是关系

关系是指存在于某些事物或实体之间的关联。在不同的领域中有不同的延伸，在哲学范畴中，关系是指反映事物及其特性之间相互联系的抽象范畴，是不同事物、特性的一种统一形式。世界上的任何事物都同它周围的事物相互联系着，这种联系表明它们彼此存在着一致性、共同性，当一种事物发生变化时，就会影响到另一种事物，从而在此基础上形成

不同的事物、特性的统一形式，即表现为一定的关系。

关系是事物相互联系的必要因素，不同关系表现着事物、特性的不同联系方式，每一种关系都是不同事物、特性的具体统一形式。事物之间的关系，以及它们特性之间的关系，是由世界物质统一性决定的。

关系是客观的，为事物所固有，存在于相应的事物之间。任何事物总是处在和其他事物的一定关系中，只有在同其他事物的关系中，它才能存在和发展，它的特性才能表现出来。事物的存在和事物的相互关系是统一的。事物的发展变化会导致该事物同其他事物原有关系的改变、消失和产生新的关系；而某个事物和其他事物关系的变化也会引起该事物的相应变化。思想中的关系是客观事物之间的关系的反映。

世界上的事物、现象以及它们的特性是复杂的、无限多样的，事物之间的关系也是复杂的、无限多样的。不同事物及其不同特性，按着各种不同类型的关系而彼此联系在一起，例如，空间与时间的关系、整体与部分的关系、原因与结果的关系、内容与形式的关系以及遗传关系、函数相依关系、内部关系与外部关系等。

不同的事物之间所具有的相同的关系的还可以构成关系集合。

在关系型数据库的设计中，是指实体(表)之间的具有某种含义的关联，而这种关联关系在关系集合中也要被唯一标识，一个可以唯一标识的关联叫做关系事件。每一个关系有一个描述关系内容、功能的名称，如经理与员工之间有关系，需要用“管理”来描述这种关系，关系是有方向的，如经理“管理”员工，而员工“服从”与经理，又是一种关系，如图 3-1 所示。

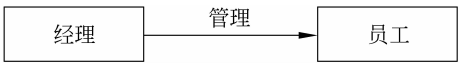

图 3-1 经理和员工关系

在关系中涉及两个重要的概念——关系的度和递归关系。

(1) 关系的度。包含于特定关系中的实体叫做参与者，在关系中参与者的数目叫做关系的度，指明了参与在一个关系中的实体的数目，度数为 1 的关系成为一元关系，通常被称为递归关系，度数为 2 的关系叫做二元关系，度数超过 2 的关系叫做复杂关系，如度数为 3 的关系叫做 3 元关系，依次类推，经常遇到的关系为二元关系，偶尔会遇到三元关系和四元关系，如图 3-2 所示。

(2) 递归关系。在不同的角色中有多次具有相同性质的实体参与的关系。如在一个班级中，有很多学生实体，但这些学生实体中有的比较特殊，有的是班长，他不但具有学生的性质，同时也可以作为班长身份管理学生，班长是学生，班长可以管理其他学生，如图 3-3 所示。

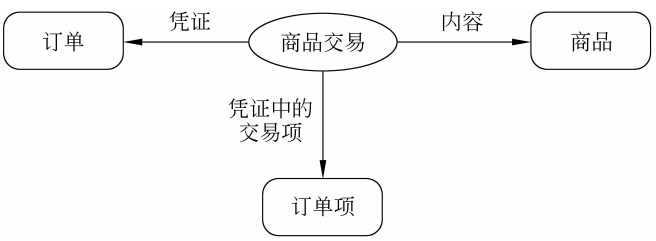

图 3-2 关系的度

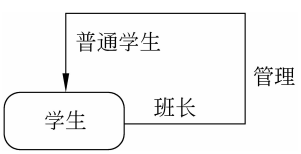

图 3-3 递归关系

在这里，班长也是学生中的一员，换句话说，学生实体参与管理关系两次，第一次作为班长参与，第二次作为一个普通学生参与。可以给关系一个角色名字以表明每个参与的实体在关系中所起的作用，角色名字对于递归关系决定参与实体的作用是非常重要的。

4. 什么是 E-R 关系图

E-R 图(Entity Relationship Diagram)称为实体-关系图，用于图形化描述实体-关系模型，是指提供了表示实体型、属性和联系的方法，用来描述现实世界的概念模型。

E-R 方法：是“实体-关系方法”(Entity-Relationship Approach)的简称。它是描述现实世界概念结构模型的有效方法。

实体-关系模型或 E-R 图是由美籍华裔计算机科学家陈品山(Peter Chen)发明，是概念数据模型的高层设计描述时所使用的数据模型或模式图，它为表述实体-关系模式使用图形化方式的数据模型提供了图形符号。这种数据模型典型的应用是在数据分析和设计的第一阶段，如在需求分析阶段用来描述信息需求以及需要存储在数据库中的信息的类型。

数据建模技术可以用来描述特定领域的任何本体(就是对使用的术语和术语之间的联系的概述和分类)。在基于数据库的应用系统设计的情况下，在后续的阶段(通常叫做逻辑设计)，概念模型要映射到逻辑模型如关系模型上，最后要在物理设计期间映射到物理模型上，有时这两个阶段可以被一起称为“物理设计”。

5. E-R 图的基本要素

通常，使用实体-关系图来建立数据模型。可以把实体-关系图简称为 E-R 图，相应地可把用 E-R 图描绘的数据模型称为 E-R 模型。E-R 图中包含了实体(即数据对象)、属性以及实体之间的关系 3 种基本成分，通常用矩形框符号代表实体，用连接相关实体的菱形框表示关系，用椭圆形或圆角矩形表示实体(或关系)的属性，并用直线把实体(或关系)与其属性连接起来。

人们通常就是用实体、属性以及关系这 3 个概念来分析、描述和设计现实问题的，因此，E-R 模型比较接近人的习惯思维方式。此外，E-R 模型使用简单的图形符号表达系统分析员对问题域的理解，不熟悉计算机技术的用户也能理解它，因此，E-R 模型可以作为用户与分析员之间有效的交流工具。

1) 实体型(Entity)

具有相同属性的实体具有相同的特征和性质，用实体名及其属性名集合来抽象和刻画同类实体；在 E-R 图中用矩形表示，矩形框内写明实体名；如用户张三、用户李四都是实体。

2) 属性(Attribute)

实体所具有的某一特性，一个实体可由若干个属性来刻画。在 E-R 图中用椭圆形表示，并用无向边将其与相应的实体连接起来；如用户的账号、密码、联系方式、住址等都是用来描述用户这一实体的不同性质的属性。如果是多值属性的话，在椭圆形外面再套实线椭圆。

3) 关联关系(Relationship)

数据对象彼此之间相互连接的方式称为联系，也称为关系。关系按照参与到关系中具体数据的数量，可分为以下 3 种类型。

(1) 一对一关系(1∶1)

例如，一个班级有一个班长，而每个班长只在一个班级任职，则班级与班长的关系是一

对一的。

(2) 一对多关系(1 : N)

例如,某间学校的大楼,可以包含很多教室,但其中的每个教室只能属于包含它的大楼,这样大楼与房间之间存在一对多的关系。

(3) 多对多联系(M : N)

一个学生可以学习多门课程,而每门课程可以有不同的多个学生来学习,这样学生和课程之间就是多对多的关系。关系也可能有属性特征,如学生"学"某门课程时所取得的成绩,既不是学生的属性也不是课程的属性,是存在了学习课程关系后的产物,所以成绩既属于某名特定的学生又属于某门特定的课程,它是学生与课程之间的关系"学习课程"的属性。

例如,描述一个班级、学生、课程、教师之间的 E-R 图如图 3-4 所示。

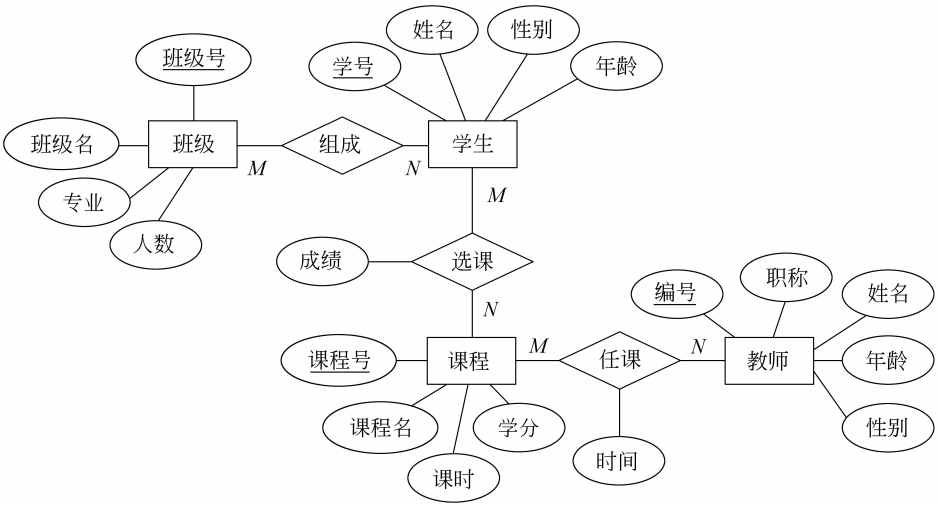

图 3-4 E-R 图实例

3.2 关系型数据的设计与优化

3.2.1 范式设计概述

数据库范式在数据库设计中的地位一直很暧昧,教科书中对于数据库范式都给出了学术性的定义,但实际应用中范式的应用却不甚乐观,后面内容会用简单的语言和一个简单的数据库 DEMO 将一个不符合范式的数据库一步步从第一范式实现到第四范式。

3.2.2 范式的目标

应用数据库范式可以带来许多好处,但是最重要的好处可以归结为以下 3 点:

(1) 减少数据冗余(这是最主要的好处,其他好处都是由此而附带的)。

(2) 消除异常(插入异常、更新异常、删除异常)。

(3) 让数据组织的更加和谐。

3.2.3 什么是范式

简单地说，范式是为了消除重复数据减少冗余数据，从而让数据库内的数据更好地组织，让磁盘空间得到更有效利用的一种标准，满足高等级范式的先决条件是满足低等级范式（例如满足第二范式一定满足第一范式才有可能成立），下面介绍各种范式。

1. 第一范式

如果一个关系模式 R 的所有属性都是不可分的基本数据项，则 $R\in$1NF。

第一范式（1NF）是指数据库表的每一列都是不可分割的基本数据项，同一列中不能有多个值，即实体中的某个属性不能有多个值或者不能有重复的属性。

2. 第二范式

若关系模式 $R\in$1NF，并且每一个非主属性都完全函数依赖于 R 的主码，则 $R\in$2NF。

第二范式（2NF）是在第一范式（1NF）的基础上建立起来的，即满足第二范式（2NF）必须先满足第一范式（1NF）。第二范式（2NF）要求数据库表中的每个实例或行必须可以被唯一地区分。为实现区分通常需要为表加上一个列，以存储各个实例的唯一标识。这个唯一属性列被称为主关键字或主键、主码。

第二范式（2NF）要求实体的属性完全依赖于主关键字。所谓完全依赖是指不能存在仅依赖主关键字一部分的属性，如果存在，那么这个属性和主关键字的这一部分应该分离出来并形成一个新的实体，新实体与原实体之间是一对多的关系。为实现区分，通常需要为表加上一个列，以存储各个实例的唯一标识。简而言之，第二范式就是非主属性非部分依赖于主关键字。

3. 第三范式

关系模式 $R<U,F>$ 中若不存在这样的码 X、属性组 Y 及非主属性 $Z(Z\ Y)$，使得 $X\rightarrow Y,Y\rightarrow Z$ 成立，则称 $R<U,F>\in$3NF。

如果 $R\in$3NF，则 R 的每一个非主属性既不部分函数依赖于候选码也不传递函数依赖于候选码。

如果 $R\in$3NF，则 R 也是 2NF。

采用投影分解法将一个 2NF 的关系分解为多个 3NF 的关系，可以在一定程度上解决原 2NF 关系中存在的插入异常、删除异常、数据冗余度大、修改复杂等问题。

将一个 2NF 关系分解为多个 3NF 的关系后，并不能完全消除关系模式中的各种异常情况和数据冗余。

4. 其他高级范式

（1）BC 范式（BCNF）

设关系模式 $R<U,F>\in$1NF，如果对于 R 的每个函数依赖 $X\rightarrow Y$，若 Y 不属于 X，则 X 必含有候选码，那么 $R\in$BCNF。

（2）第四范式

关系模式 $R<U,F>\in$1NF，如果对于 R 的每个非平凡多值依赖 $X\rightarrow\rightarrow Y(Y\ X)$，$X$ 都含有候选码，则 $R\in$4NF。

$$(X\rightarrow Y)$$

如果 $R \in 4NF$，则 $R \in BCNF$。

不允许有非平凡且非函数依赖的多值依赖。

允许的是函数依赖（是非平凡多值依赖）。

3.2.4 范式的 Power Designer 操作

让我们先从一个未经范式化的表看起，如图 3-5 所示。

先对表作简单说明。其中，employeeId 是员工号，departmentName 是部门名称，job 代表岗位，jobDescription 是岗位说明，skill 是员工技能，departmentDescription 是部门说明，address 是员工住址。

1. 对表进行第一范式（1NF）

如果一个关系模式 R 的所有属性都是不可分的基本数据项，则 $R \in 1NF$。

简单地说，第一范式就是每一个属性都不可再分。不符合第一范式则不能称为关系数据库。对于图 3-5 中的表，不难看出 Address 是可以再分的，例如“北京市××路××小区××号”显然不符合第一范式，对其应用第一范式则需要将此属性分解到另一个表，如图 3-6 所示。

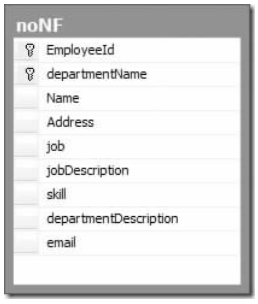

图 3-5 未经范式化的表

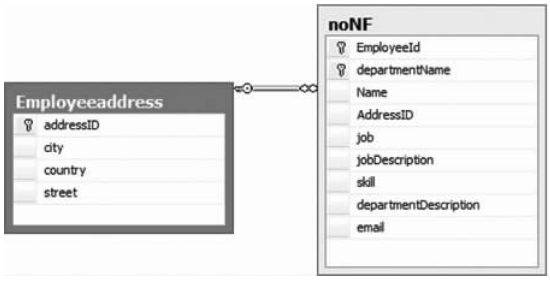

图 3-6 对表进行第一范式

2. 对表进行第二范式（2NF）

若关系模式 $R \in 1NF$，并且每一个非主属性都完全函数依赖于 R 的码，则 $R \in 2NF$。

简单地说，表中的属性必须完全依赖于全部主键，所以只有一个主键的表如果符合第一范式，那一定是第二范式，而不是部分主键。这样做的目的是进一步减少插入异常和更新异常。在图 3-6 的表中，departmentDescription 是由 DepartmentName 所决定，但却不能由 EmployeeId 决定，故要 departmentDescription 对主键是部分依赖，对其应用第二范式的表如图 3-7 所示。

3. 对表进行第三范式（3NF）

关系模式 $R<U,F>$ 中若不存在这样的码 X、属性组 Y 及非主属性 $Z(Z \subseteq Y)$，使得 $X \to Y, Y \to Z$，成立，则称 $R<U,F> \in 3NF$。

简单地说，第三范式是为了消除数据库中关键字之间的依赖关系，在上面经过第二范式化的表中，可以看出，jobDescription（岗位职责）是由 job（岗位）所决定，这样 jobDescription 依赖于 job，可以看出这不符合第三范式，对表进行第三范式后的关系如图 3-8 所示。

图 3-7　对表进行第二范式

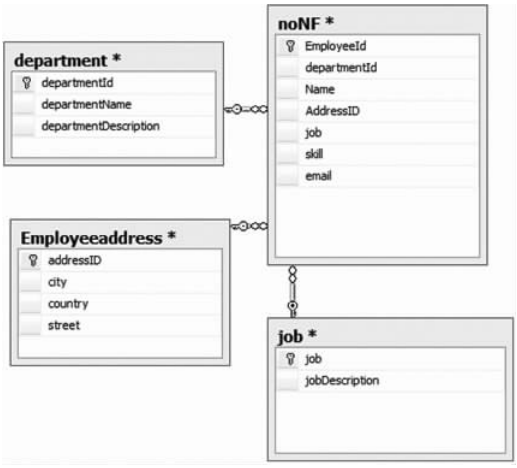

图 3-8　对表进行第三范式

图 3-8 的表中，已经不存在数据库属性互相依赖的问题，所以符合第三范式。

4. 对表进行 BC 范式（BCNF）

设关系模式 $R<U,F>\in 1NF$，如果对于 R 的每个函数依赖 $X\rightarrow Y$，若 Y 不属于 X，则 X 必含有候选码，那么 $R\in BCNF$。

简单地说，BC 范式是在第三范式基础上的一种特殊情况，即每个表中只有一个候选键（在一个数据库中每行的值都不相同，则可称为候选键），在上面第三范式的 noNf 表中可以看出，每一个员工的 email 都是唯一的（难道两个人用同一个 email），则此表不符合 BC 范式，对其进行 BC 范式化后的关系如图 3-9 所示。

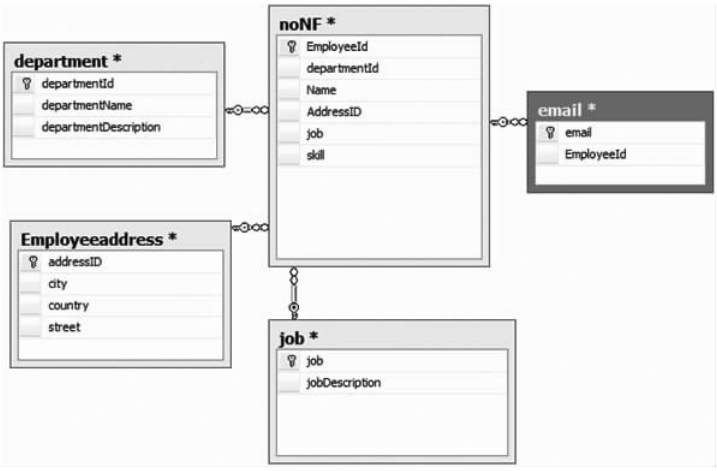

图 3-9　对表进行 BC 范式

5. 对表进行第四范式（4NF）

关系模式 $R<U,F>\in 1NF$，如果对于 R 的每个非平凡多值依赖 $X\rightarrow\rightarrow Y(Y\subseteq X)$，$X$ 都含有候选码，则 $R\in 4NF$。